Hybridization of Deep Learning Models for Multiclass Attack Detection in Wireless Sensor Networks

Omar Almomani [1], Areen M. Arabiat [2,*], Al-Ahmed Hind [3], and Esraa Alsariera [4]

[1] Department of Networks and Cybersecurity, Faculty of Information Technology, Al-Ahliyya Amman University, Amman, Jordan
[2] Department of Communications and Computer Engineering, Faculty of Engineering, Al-Ahliyya Amman University, Amman, Jordan
[3] Faculty of Education, Imam Mohammad Ibn Saud Islamic University (IMSIU), Saudi Arabia
[4] Department of Computer Science, Faculty of Information Technology, Al-Ahliyya Amman University, Amman, Jordan
Email: o.almomani@ammanu.edu.jo (O.A.); a.arabiat@ammanu.edu (A.M.A.); hmaamad@imamu.edu.sa (A-A.H.); e.alsariera@ammanu.edu.jo (E.A.)
*Corresponding author

***Abstract*—Flooding, blackhole, and forwarding are Wireless Sensor Networks (WSNs) attack types that have a severe impact on the reliability of the network and the integrity of data. This paper reports on a detailed benchmarking and hybridization analysis of seven deep learning models regarding multiclass intrusion detection on the WSNBFSF dataset. Comparison is made of the Artificial Neural Network (ANN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and their mixtures, CNN-RNN, CNN-LSTM, and RNN-LSTM. The same experimental conditions were used to train all the models and make the evaluation fair. The experimental findings are indicative of the fact that the ANN has the highest performance with accuracy, precision, recall, and F1−Score of 99.81% in split validation and 99.89% in 5-fold cross-validation, offering close to perfect discrimination of all attack types. The LSTM (98.27%) and RNN-LSTM (97.78%) memory augmented models also demonstrate a high ability to capture traffic time dependencies in WSNs, whereas the CNN-based models are efficient in spatial pattern extraction but slightly worse on fine-grained attack detection. Conversely, a standalone RNN has problems of vanishing gradients and thus gives an accuracy of 84.77 percent. In summary, the results provided serve as a pointer to architecture-specific strengths, and they provide important information in designing a scalable, real-time, and resource-efficient deep learning-based intrusion detection system with the peculiarities of WSN environments.**

***Keywords*—wireless sensor networks, deep learning, flooding attack, blackhole attack, forwarding attack, WSNBFSF dataset**

Manuscript received August 25, 2025; revised September 16, 2025; accepted September 23, 2025; published January 9, 2026.

I. Introduction

Wireless Sensor Networks (WSNs) are an inexpensive and simple way for sensing devices to communicate with their environment. They are extensively utilized in many domains, including surveillance, industry, security, health, and the military. Because they lack specific routing protocols, WSN devices that run on limited-life batteries must be actively operated to avoid rapid battery depletion when in sleep mode, which can result in vampire attacks. To transfer data packets, WSNs need to install sensor nodes at various depths and use a routing mechanism. These devices are exposed to huge populations since they are lightweight, portable, and low-power [1−4]. The importance of WSNs for essential applications is growing, yet network intrusions and sensor failures present increasing risks to WSN security and dependability. But their use in open environments exposes them to cyber threats, compromising data integrity, confidentiality, and availability [5−7]. However, Large-scale WSNs are especially susceptible to environmental damage, cyberattacks, and physical manipulation since they have hundreds or thousands of sensor nodes. Conventional anomaly detection techniques have trouble adjusting to large-scale sensor deployments and changing conditions in the environment [8−10]. With the widespread usage of CNN and RNN for processing different kinds of data, deep learning has made considerable strides. When it comes to translation invariance, CNN is a feedforward neural network, and RNN is a time recurrent neural network that works well with time series data. Time series dependencies can be processed and predicted more accurately with the LSTM (Long and Short-Term Memory) RNN structure [11] .With the use of algorithms and learning models, Machine Learning (ML) optimizes the process by gradually identifying patterns in data collection. It has been applied to several domains, such as speech recognition, bioinformatics [12−14], and WSNs. For the interpretation of datasets that the human brain is unable to process, machine learning is crucial. On the other side, machine learning algorithms can assist in the administration of IoT nodes in smart cities for uses such as building automation, garbage management, and traffic monitoring. However, bandwidth, energy consumption, coverage, and connectivity provide design issues for these networks. The unpredictability of data streams and the mobility of sensor nodes, which might result in changes in network architecture owing to dynamic changes in WSN environments, make machine learning essential for WSN applications [15, 16].

The application of machine learning, especially anomaly detection, for reducing costs in the security industry is growing. This method has demonstrated potential in fields such as packet analysis, eliminating Denial-of-Service (DoS) attacks [17−19], and improving network availability. Security is crucial in some Wireless Sensor Networks (WSNs) that handle private information in potentially dangerous settings. Confidentiality, authenticity, integrity, and novelty of data are important security issues that require strong security protocols [17, 20−22]. In intelligent cities, IoT functions like trash management, traffic monitoring, and healthcare can be managed by AI and machine learning. However, issues including web range, connectivity, power consumption, bandwidth needs, and infrastructure development are problems for WSN-based IoT. Because the sensor nodes in WSNs are autonomous and constrained by their capabilities, research focuses on enhancing network lifetime and quality of service [23]. These problems may be solved with the aid of ML techniques [24, 25]. However, Artificial Neural Networks (ANNs) with several layers are used in Deep Learning (DL), a subset of machine learning. Unlike conventional algorithms, its performance becomes better with more data entered. As a result of technological improvements that enable small training and deployment in gadgets like smartphones, DL has grown in popularity. It is difficult for statistical and traditional machine learning time series forecasting algorithms to derive information from nonlinear patterns and data. LSTMs, GRUs, RNNs, and 1D-CNNs are a few architectures that work well for learning from sequential data [26]. Fig. 1 shows the major WSN components.

The purpose of this research is to use deep learning techniques, specifically Long Short-Term Memory (LSTM), Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and a hybrid of each of them, to create an advanced detection system for WSNs attacks. The novelty is in combining these disparate architectures to improve temporal pattern recognition and feature extraction, leading to it being possible to accurately detect and categorize different kinds of attacks. The study aims to increase the reactivity to new threats by concentrating on real-time data processing capabilities, which would greatly improve the safety and reliability of WSNs.

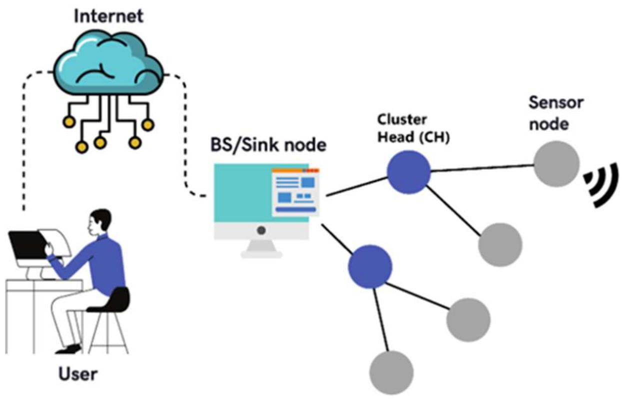


Fig. 1. WSN component [26].

The rest of this paper will be organized as follows. In Section II, the background literature is reviewed, and major gaps fueling this research are indicated. In Section III, the section dealing with the methodology details the WSNBFSF dataset, preprocessing, and DL classification algorithms. Section IV is a discussion of the findings. Lastly, the paper ended with Section V, which indicated the future work and conclusion.

II. Related Works

Prior research has extensively explored WSNs attack detection using ML and DL, establishing foundational insights into binary classification (e.g., normal vs. malicious), while leaving critical gaps in multiclass attack classification. The most relevant studies of WSN attack detection using ML and DL will be discussed in this section.

Using different Deep Learning techniques, the study investigates common WSN attack detection. The WSN Dataset and WSN-BFSF datasets are used to evaluate both soft and hard ensemble methods. Overall accuracy is increased by mixing forecasts from many models, according to the results. The accuracy of the hard ensemble technique was 99.967% on the WSN-DS dataset, whereas the soft ensemble technique was 98.12% [27]. By combining a hybrid CNN-LSTM with an attention mechanism, the suggested model in Ref. [28] enhanced feature extraction and pattern recognition, hence boosting IoT intrusion detection. Real-time feature selection optimization by the dynamic PSO algorithm improves classification precision and flexibility in response to changing threats. High accuracy rates of 98.73% with CNN, 99.87% with LSTM, 99.12% with CNN-LSTM, and 98.88% with enhanced CNN-LSTM with attention are demonstrated by experimental assessments.

Tolba [29] introduced a novel Intrusion Detection System (IDS) model that combines machine learning and deep learning methods for wireless sensor networks. The suggested hybrid model successfully captures the temporal correlations in the data by extracting features using an LSTM network. Its structure consists of a layer of dropouts for regularization, a dense layer, and an LSTM layer. To detect and categorize four different kinds of Denial of Service (DoS) attacks, blackhole, Grayhole, flooding, and scheduling, the study additionally combines LSTM with

Random Forest. With accuracy, precision, recall, F1−Score, and AUC scores of 0.996, 0.98, 0.98, 0.98, and 0.99, respectively. Using a multi-criteria approach, Quayed *et al.* [24] suggested a framework to improve cybersecurity in Industry 4.0 WSNs. It employs deep learning and machine learning methods to detect and stop intrusions. Three models Decision Tree, MLP, and Autoencoder—are used by the framework. While the MLP model achieves 99.52% accuracy, the Decision Tree model achieves 99.48% accuracy. The accuracy of the binary classification Autoencoder model is 91%. When compared to benchmark models like Random Forest and Logistic Regression, the suggested framework performed better.

Altameemi *et al.* [30] used machine learning and deep learning technologies to offer a model-based cyberattack detection system. To determine attack types, it employs Recurrent Neural Network (RNN) techniques and feature reduction. After dimensional reduction and optimization, the system reaches 97% accuracy, which aids with routing protocol design and lowers the probability of attacks in Underwater Wireless Sensor Networks (UWSN) communication. On the other hand, Reddy *et al.* [31] suggested a feed-forward ANN model with a Multi-Layer Perceptron (MLP) framework that has been preprocessed using WEKA. According to experiments, the MLP performs better than the RBMC-IDS and Hoeffding adaptive trees, with 98% accuracy compared to 96.33% and 97%, respectively.

To identify and categorize DoS intrusion attacks as Flooding, Blackhole, Normal, TDMA, or Grayhole, Salmi *et al.* [32] employed a CNN-LSTM network. The dataset is a computer-generated wireless sensor network-detection system. The wireless sensor network environment is simulated using the Network Simulator NS-2, which produces 23 features for categorizing sensor states and five types of DoS attacks. The accuracy, precision, and recall scores of the CNN-LSTM model are 0.944, 0.959, and 0.922, respectively, after evaluation across 25 epochs. Using the NSL-KDD dataset, Ramkumar *et al.* [2] presented a modified Recurrent Neural Network with Long Short-Term Memory (RNN-LSTM) for IDS in WSN classification. Thorough performance validation yields exceptional results, such as 99.95% accuracy, 99.93% precision, and 95.69% robust recall, with an F1−Score of 99.80%, exceeding the standards set by present methods.

Sivakumar *et al.* [33] presented a novel Recurrent Crossover-based Dynamic Differential (RC-DD) algorithm for effective attack detection and localization in Wireless Sensor Networks (WSN). For precise sensor node location, the model employs a weighted K-Nearest Neighbor (KNN) with Mahalanobis distance. Dynamic Differential Annealed Optimization (DDAO) adjusts the hyperparameter of the Recurrent Neural Network (RNN), which detects multiple intrusions. With an accuracy rating of 98.9%, the RC-DD framework minimizes energy usage and localization inaccuracy. Moundounga and Satori [34] suggested system trains HMMs and GMMs using iterative machine learning Expectation-Maximization and reduces dimensionality in the WSN dataset using Principal Component Analysis. According to experiments, the setup consisting of three HMMs and four GMMs attains a remarkable accuracy of 94.55%. With accuracies between 0.96 and 0.98, the system performs better in cross-validation than popular machine learning classifiers. Table I summarizes the previous research studies.

TABLE I. COMPARISON OF PREVIOUS RESEARCH

Study	Model	Performance Metrics
[2]	RNN-LSTM	Accuracy: 99.95%, Precision: 99.93%, Recall: 95.69%, F1−Score: 99.80%
[34]	HMMs and GMMs	Accuracy: 94.55%
[24]	MLP, Decision Tree, Autoencoder	MLP: Accuracy: 99.52%, Decision Tree: 99.48%, Autoencoder: 91%
[27]	Hard Ensemble, Soft Ensemble	Hard Ensemble: Accuracy: 99.967%, Soft Ensemble: 98.12%
[28]	Hybrid CNN-LSTM with Attention	Accuracy: 98.88% (Enhanced CNN-LSTM), 99.12% (CNN-LSTM), 99.87% (LSTM), 98.73% (CNN)
[29]	LSTM + Random Forest	Accuracy: 0.996, Precision: 0.98, Recall: 0.98, F1−Score: 0.98, AUC: 0.99
[30]	RNN	Accuracy: 97%
[31]	MLP, RBMC-IDS, Hoeffding Adaptive Trees	MLP: Accuracy: 98%, RBMC-IDS: 96.33%, Hoeffding: 97%
[35]	CNN-LSTM	Accuracy: 0.944, Precision: 0.959, Recall: 0.922
[33]	RC-DD with KNN	Accuracy: 98.9%

III. METHODOLOGY

This section provides an outline of the methodological framework adopted to conduct the study. Hence, the section focuses on the dataset, preprocessing of data, model building, and evaluation procedures. Fig. 2 shows the methodological framework.

A. WSNBFSF Dataset

The WSNBFSF dataset [36] provides a thorough analysis of WSN traffic and contains 312,106 attacks, which are categorized into four groups: normal traffic, floods, blackhole attack traffic, and selective forwarding attack traffic. Network congestion and energy resource strain are caused by black holes, which also act as portals for other attacks, such as sinkholes or selective forwarding. Researchers studying network security frequently encounter these attacks, which are a subset of DDoS attacks. Finding and differentiating between vectors of attack and regular network activity is its main goal. This information, which comes from accurate preprocessing, improves our knowledge of network vulnerabilities by offering insightful information about the dynamics of cyber threats and vulnerabilities in WSNs [37, 38]. Fig. 3 shows the WSNBFSF dataset target distribution. And Table II shows the features list of WSNBFSF dataset.

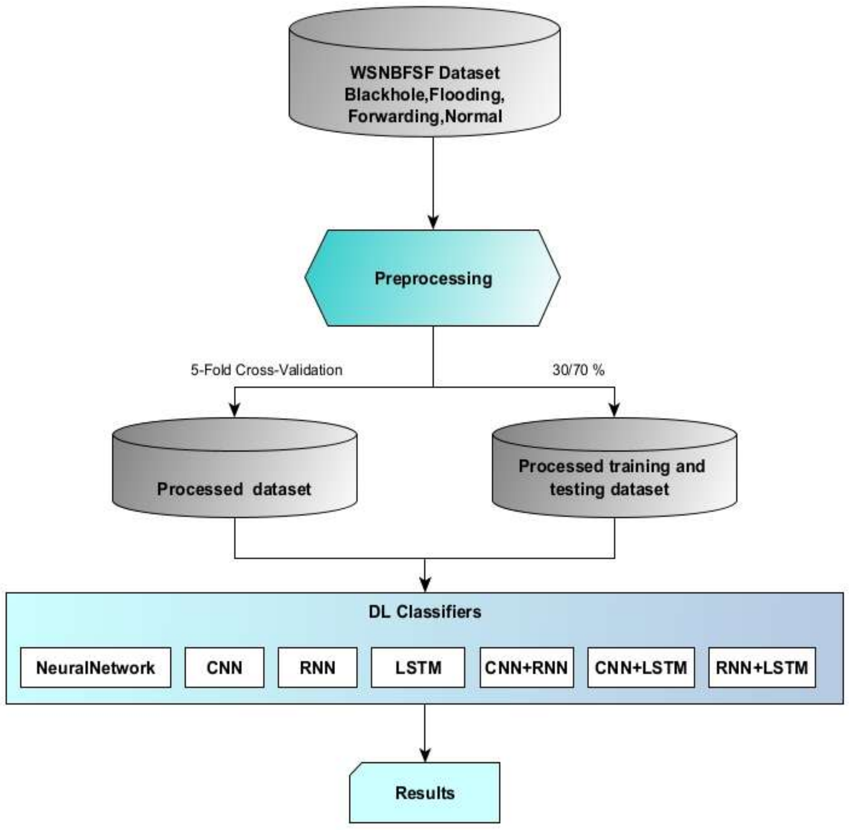


Fig. 2. Methodological framework.

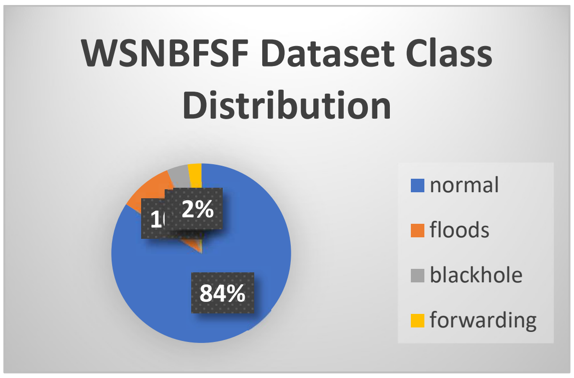


Fig. 3. WSNBFSF dataset class distribution.

TABLE II. FEATURES LIST OF WSNBFSF DATASET

Number	Features	Features Description
1	Event	1 (sending), 2 (receiving), 3 (forwarding), 4 (dropping), and 5 (Energy information).
2	Time	The time of the event performed in the row
3	S_Node	Source node number
4	Node_id	The node number of the relevant node
5	Rest_Energy	The remaining energy of the relevant node
6	Trace_Level	represents the level recorded by a sensor at a specific
7	Mac_Type_Pckt	MAC type of the packet
8	Source_IP_Port	Port number of the source node
9	Des_IP_Port	Port number of the Destination node
10	Packet_Size	Forwarded packet size
11	TTL	The lifetime of the forwarded traffic in the network
12	Hop_Count	Number of nodes passed
13	Broadcast_ID	The ID number of the broadcast packets
14	Dest_Node_Num	ID of the target node
15	Dest_Seq_Num	Sequence number of the traffic forwarded to the destination
16	Src_Node_ID	Source node ID number
17	Src_Seq_Num	Source sequence number of traffic forwarded to destination
18	Class	Blackhole, Flooding, Forwarding, and Normal

B. Preprocessing

The WSNBFSF dataset was employed to test the proposed classification framework. This dataset comprises multiple classes that display different networking behaviors, including normal traffic and different categories of attacks (floods, blackhole, and forwarding). A high imbalance across classes was observed, with potential adverse consequences on the model's performance and predictive bias toward majority classes. To solve this problem, the following steps were used to enhance classification performance.

1) Cleaning and transformation of the dataset

First, the dataset was examined for missing and inconsistent values. Instances containing null or undefined values were either imputed with means of numerical features or removed if the number of cases with missing values was marginal. Label encoding for non-numeric features was done to make the data compatible with machine learning algorithms.

2) Feature scaling

To construct a feature space that is standard and stable for faster model convergence, all input features were scaled using StandardScaler to have a zero mean and unit variance. This normalization is paramount for distance-based algorithms and for deep learning.

3) Handling class imbalance

The original sample possessed a highly imbalanced distribution, in which normal instances far exceeded the instances in attack classes. To maintain equal representation from all the classes and to avoid bias to the majority classes by the models, a hybrid resampling technique was applied. Utilizing the Random Under Sampler and Random Over Sampler classes in Python, the number of normal was under-sampled to 29,844, whereas all attack classes were over-sampled to 29,844. With this setting, all the classes will have an equal share of representation during model training, which will improve classification for all categories and enhance the generalization of the classes. Fig. 4 shows the WSNBFSF dataset target distribution after balancing.

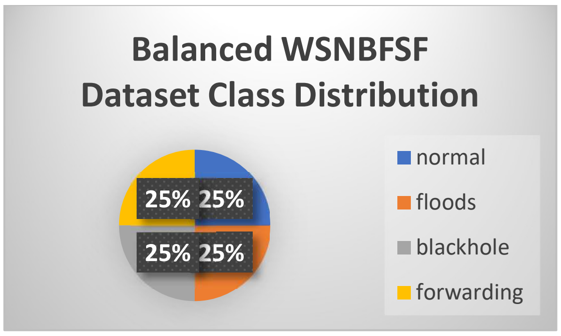


Fig. 4. Balanced WSNBFSF dataset class distribution.

4) Train-test validation

To ensure the evaluation rigor, the final processed dataset applied a train-test split of 70:30 and 5-fold cross-validation. A stratified split was to ensure that class distributions in all subsets were not distorted by first splitting the dataset into training, validation, and test sets. To prevent data leakage, resampling methods were performed on the training set and validation, and test sets were left as they were. This guarantees that model analysis is carried out on unrewarded information, showing the actual class distribution. Also, a duplicate check was done on the splits to ensure that there were no duplicate instances in the training and evaluation sets.

C. DL Classification Algorithms

1) Neural network (multi-layer Perceptron classifier)

MLP is a neural network-based machine learning technique that combines significant amounts of data and multi-layered Artificial Neural Networks (ANN) to simulate human brain neurons [39]. It is recognized for its dependability because of its approach of directional learning. However, an ANN is made up of one or more hidden layers, an input layer, and an output layer, as shown in Fig. 5. It has several basic unit neurons that conduct layer-by-layer conduction between neurons to represent data. If there is more than one hidden layer, the output of the hidden layer is added to the next one, and each layer after that is summed using a different weight. After determining the ideal weight for inputs and biases using a typical Sigmoid function, the method enters this information into the activation level via the transfer function to generate output [40−42].

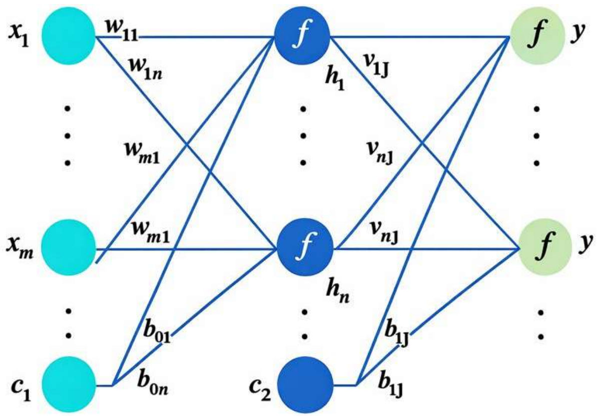


Fig. 5. Structure of ANN classifier [42].

2) Convolutional Neural Networks (CNNs)

Are deep learning models that are adaptable and multipurpose since they can learn hierarchical picture features. Their adaptive feature determination from raw input data makes them especially well-suited for recognition of features and data informativeness evaluation. CNNs can be trained on big datasets like ImageNet, and capture both low-level and high-level semantic information [43]. With its effective design that combines weight sharing, down-sampling, and local perceptual fields to detect complex spatial patterns, CNN is a potent tool in image recognition, computer vision, and fault detection. Convolutional layers are used for feature detection, pooling layers are used to reduce dimensionality, and fully connected layers are used for classification [44].

3) Recurrent Neural Networks (RNNs)

Are deep learning that handles time series data by identifying contextual correlations in sequential data. Time series modeling is made possible by RNNs' recurrent mechanism, which affects their output in contrast to typical feedforward neural networks [45]. However, RNNs are suitable for time-series studies because of their capability to process sequential input and use feedback connections to learn previously computed sequences [46].

4) Long Short-Term Memory (LSTM)

To mitigate the vanishing gradient problem prevalent in Recurrent Neural Networks (RNNs), Hochreiter and Schmid Huber introduced Long Short-Term Memory (LSTM) networks [47, 48]. LSTMs employ sophisticated gating mechanisms to regulate the flow of information, thereby facilitating the retention of long-term dependencies. As an enhanced variant of RNNs, LSTMs incorporate a memory unit designed for the dynamic updating and storage of information. This architecture features three distinct gate units: input gates, forget gates, and output gates. These gates function to selectively manage the retention, discarding, and retrieval of information, thus empowering LSTMs to effectively learn from both short-term and long-term dependencies in sequential data [49, 50].

5) Hybrid CNN-RNN

The hybrid combination of CNNs and RNNs is, therefore, designed to realize extracting both spatial patterns and temporal dependencies in WSN data [51]. In essence, architecture emphasizes CNN layers to extract local features of the data, wherein the recurrent layer ingrains the sequential dependency for multiclass classification output. The network architecture begins with a stack of two 1-D convolution layers, 64 filters each, with the kernel size 3 and padding 'same'. The `relu` activation introduces nonlinearity. BatchNormalization follows each convolution layer to stabilize and speed up the training process. These layers identify low-level local patterns in the input features space. This is followed by MaxPooling1D pooling with a pool size of 2, which down-samples the input feature maps and reduces computational cost while retaining important features. Then comes the deeper convolution part: two more convolutional layers, each with 128 filters, kernel size of 3 again, and batch normalization-this set allows the model to learn more abstract, high-level representations. One more MaxPooling1D and Dropout with a dropout rate of 0.4 were introduced to avert overfitting. After spatial feature extraction, the post-processed features are fed into a SimpleRNN with 128 units. This layer captures temporal dependencies and sequential behaviors occurring in WSN traffic data. Setting the parameter return_sequences to False passes only the last hidden state to the dense layers, which essentially summarizes the entire input sequence. A second Dropout layer, again with a dropout rate of 0.4, is applied for overfitting prevention. The last part of the architecture consists of two fully connected layers: a 128-unit layer with ReLU activation followed by Dropout at a rate of 0.3, and the output layer with four units and softmax activation for the four classes of the classification task (say, Normal, Flooding, Blackhole, Forwarding). Adam optimizer with a learning rate of 0.001, compiles the model, which is then trained using the sparse categorical cross-entropy loss function, which is appropriate for integer-labeled multiclass targets. The performance of the network is evaluated using accuracy.

6) Hybrid CNN-LSTM

To enhance the spatial and temporal characteristics of features in WSN traffic data, CNN-LSTM incorporates a hybrid model [35]. Architecture learns discriminative local patterns through deep convolutional layers and LSTM layers to grasp sequential dependencies. This offers excellent capabilities for WSN anomaly or attack detection based on multiclass classification. Temporal features are obtained, which are finally passed down to an LSTM layer with 128 hidden units. The LSTM is configured with return_sequences=False, so that only the hidden state of the last time step is passed to the next layer. This hidden state is meant to summarize the temporal dynamics of the input sequence. To promote further regularization, a Dropout layer with a 40 percent dropout rate was added to the LSTM output. The LSTM output passes through a fully connected layer with 128 neurons and ReLU activation, followed by a Dropout layer with a 30 percent dropout rate. Finally, the output layer uses a dense layer with 4 neurons with softmax activation to represent the four classes for the classification task. The model is compiled using the Adam optimizer with a learning rate of 0.001. The sparse categorical cross-entropy loss function is used to handle integer-encoded target labels for multiclass classification. Model performance is evaluated using the accuracy metric.

7) Hybrid RNN-LSTM

A hybrid Recurrent Neural Network–Long Short-Term Memory (RNN-LSTM) model was built to model short-term and long-term temporal patterns in WSN traffic sequence data [52]. The architecture is a better fit for gleaning sequential dependencies and behavioral signatures associated with normal and anomalous WSN behaviors. The first layer is a simple RNN that has 128 units of recurrent; this layer receives the input sequence one by one through time, deriving potential short-term dependencies, and transferring this contextual information across time steps. The parameter return_sequences=True is set in order for the layer to output the whole sequence of hidden states to the ensuing LSTM layer. Then, the BatchNormalization layer that stabilizes training and normalizes the activations is followed. A Dropout layer process is next in the sequence, with the 0.3 dropout rate, preventing overfitting and aiding the model to generalize better. The second recurrent layer of the model is an LSTM layer, which has 128 units. LSTMs are built to model long dependencies in sequential data; therefore, they are useful when temporal relations span more than a couple of time steps. Return_sequences = False ensures that only the final hidden state is moved on to the dense layers, encapsulating a temporal representation of the entire input sequence. BatchNormalization and Dropout with a rate of 0.4 were applied to the LSTM layer to thwart overfitting. The output of the LSTM layer passes to the fully connected dense layer consisting of 128 neurons, activated by ReLU for nonlinear transformations and abstracting at a high level. Afterward, another Dropout layer that has a 0.4 dropout rate is dropped in to minimize overfitting. Finally, the four-neuron softmax-activated dense output layer of the final stage. It corresponds to classification into four output classes (Normal, Blackhole, Flooding, Forwarding). The compilation of the model is done with Adam as the optimizer, having a learning rate of 0.001, while the loss function being optimized is sparse categorical cross-entropy, which fits multi-class classification problems with integer-encoded labels. The performance of the model is kept track of, using accuracy as the evaluation metric.

IV. Experimental and Results

A. Experiments Setting

All experiments were tested using Python and run on an i7-1065G7 processor at 1.50 GHz and 16.0 GB of RAM. Table III. shows the DL classifier parameter settings.

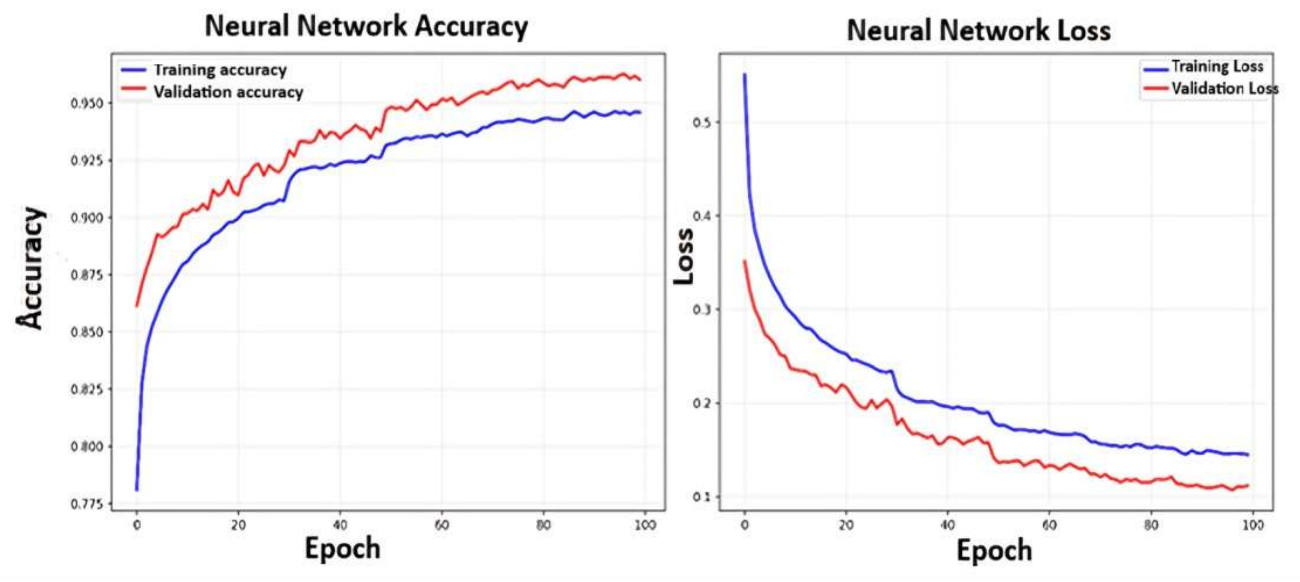


Fig. 6. Learning curves of ANN.

TABLE III. Classifier Parameters Setting

Classifier	Setting	
Neural Network (MLP)	hidden_layer_sizes	(256,128, 64)
	activation	Relu
	solver	adam
	Alpha	1e-4
	learning_rate	adaptive
	max_iter	2000
	epochs	100
CNN	hidden_layer_sizes	(64,64,128)
	Activation	ReLU (hidden), Softmax (output)
	Solver	Adam
	Learning rate	0.001
	epochs	100
RNN	hidden_layer_sizes	(64,128,128)
	Activation	ReLU (hidden), Softmax (output)
	Solver	Adam
	Learning rate	0.001
	epochs	100
LSTM	hidden_layer_sizes	(128,64,64)
	Activation	ReLU (hidden), Softmax (output)
	Solver	Adam
	Learning rate	0.001
	epochs	100
CNN+RNN	hidden_layer_sizes	CNN layers: (64, 64, 128, 128) , Simple RNN layer: 128 Dense(128)
	Activation	CNN, Dense layer=relu Output layer =softmax
	Solver	Adam
	Learning rate	0.001
	epochs	100
CNN+LSTM	hidden_layer_sizes	CNN layers: (64, 64, 128, 128) , LSTM layer: 128 , Dense(128)
	Activation	CNN layers, Dense layer=relu Output layer → =softmax
	Solver	Adam
	Learning rate	0.001
	epochs	100
RNN+LSTM	hidden_layer_sizes	RNN layers: 128 , LSTM layer: 128 , Dense (128)
	Activation	Dense layer=relu Output layer → =softmax
	Solver	Adam
	Learning rate	0.001
	epochs	100

Fig. 6 shows the learning curves of ANN model. The accuracy curves (left) indicate that both the training and the validation accuracy curve are growing steadily by epochs, and they are ultimately stabilized at 95 percent and 96 percent respectively. Notably, the accuracy of validation is always higher than the accuracy of training, which indicates that the model is not overfitted but generalizes. In the same manner, the loss curves (right) show that the training and validation loss decrease continuously, and neither of the curves diverges. This consistency between training and validation performance proves that ANN attains consistent learning behavior and strong generalization to unknown data. Thus, the fact that such learning curves are included helps us to state that the ANN model is not overfitted but instead well-regularized and can be practically deployed. In general, the findings confirm the stability of the training process and prove that the ANN is the best to be used in attaining high accuracy and minimizing the error rates. Such good results confirm the appropriateness of the model to be used in practice, where reliability and generalization are essential.

B. Performance Evaluation Metrics

The efficiency level of the DL model is assessed using the following metrics. These metrics include accuracy, recall, F-measure, and precision [39, 53]. Fig. 7 shows the confusion matrix of 4 classes classification.

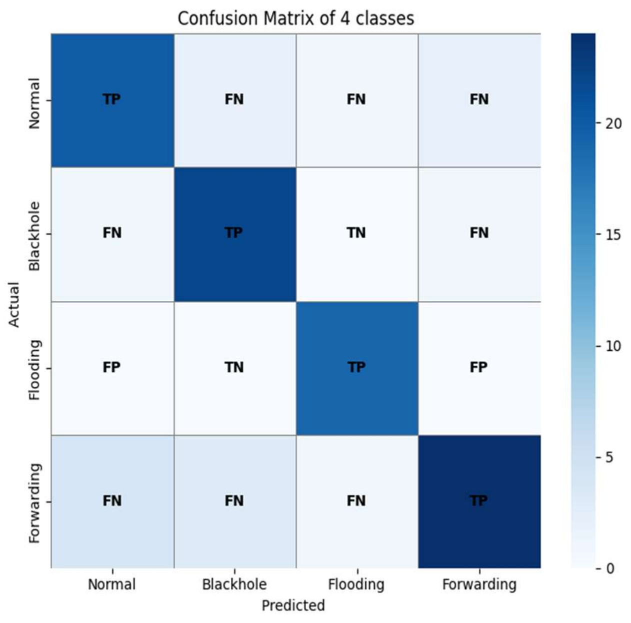


Fig. 7. Confusion matrix [54].

TP (True Positive): Represents instances accurately classified as positive, FP (False Positive): Represents instances incorrectly classified as positive, TN (True Negative): Represents instances correctly classified as negative, and FN (False Negative): Represents instances incorrectly classified as negative [41, 55, 56].

Accuracy: The proportion of correctly identified samples to all samples is known as accuracy. While accuracy is a valuable metric, it may disproportionately favor the dominant class, making it insufficient for imbalanced datasets [57, 58].

Accuracy = TP / total number of samples

Recall: is a machine learning statistic that assesses a model's capacity to accurately detect every positive instance in a dataset [59, 60].

$$\text{Recall} = \text{TP} / (\text{TP}+\text{FN})$$

Precision: indicates the proportion of well-classified positive data that is actually predicted, which means fewer false positives result from high precision [39, 61].

$$\text{Precision} = \text{TP} / (\text{TP}+\text{FP})$$

F-measure: calculates the harmonic mean of the accuracy and sensitivity rates [62].

F1-measure = 2 × (Precision × Recall) / (Precision + Recall))

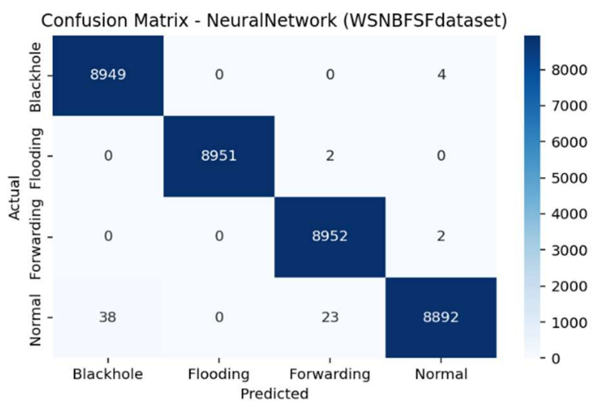
C. Classification Result (70/30 split Validation)

The WSN attacks are assessed using Neural Network, CNN, RNN, LSTM, CNN+RNN, CNN+LSTM, and RNN+LSTMML classifiers I term of accuracy, precision, recall, and F1−Score. The results of the classification experiments are presented in Table IV. Fig. 8 shows the Confusion Matrix obtained for each DL classifier.

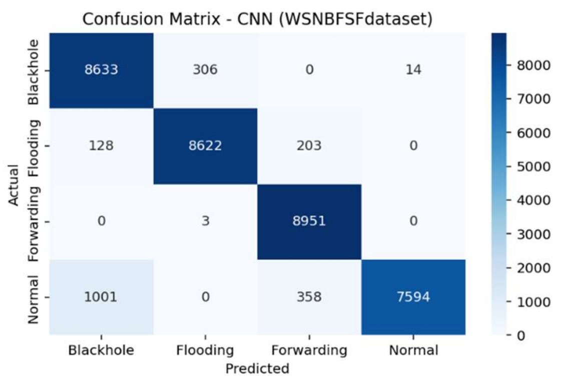
The accuracy comparison, as shown in Fig. 9, proves that the Neural Network almost attained 100% accuracy, practically perfect accuracy, thus asserting that the model is very capable of grasping the data patterns prevailing in the dataset. LSTM and RNN+LSTM follow closely below with accuracies just under 98%, hence proving the fact that memory in temporal sequences produces better classification results. CNN, CNN+RNN, and CNN+LSTM attained accuracies in the 95% range, indicating good results, though a tad inferior to their LSTM-hybrid counterparts. RNN recorded a very low performance with an accuracy of nearly 85%, proving that though one can process sequential data through recurrent mechanisms, a mere RNN might not be very useful due to vanishing gradient, unlike LSTM or hybrid networks. Generally, the outcomes prove that the deep learning models with memory seem to dominate their simpler recurrent and convolutional rivals on this task.

TABLE IV. OBTAINED RESULTS (70/30 SPLIT VALIDATION)

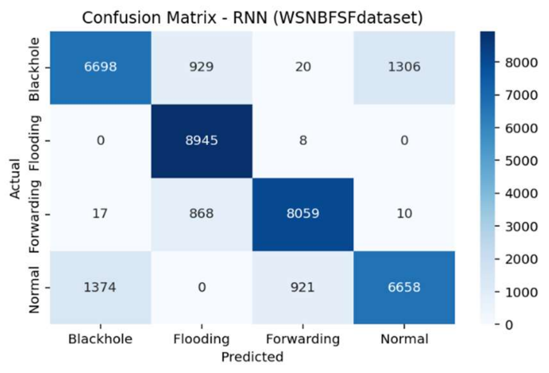
DL Classification Model	Accuracy	Precision	Recall	F1−Score
Neural Network	99.81	99.81	99.81	99.81
CNN	94.38	94.72	94.38	94.33
RNN	84.77	84.76	84.77	84.46
LSTM	98.27	98.33	98.27	98.27
CNN+RNN	95.17	95.46	95.17	95.16
CNN+LSTM	94.93	95.22	94.93	94.93
RNN+LSTM	97.78	97.89	97.78	97.78



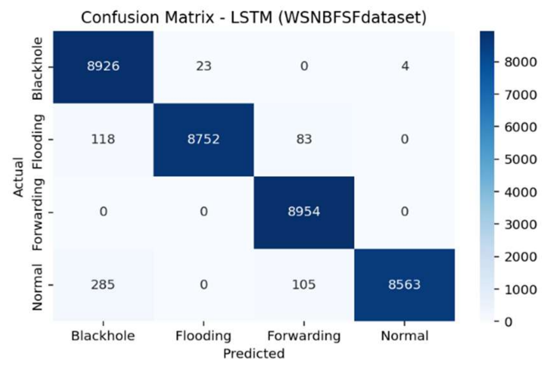
(a)



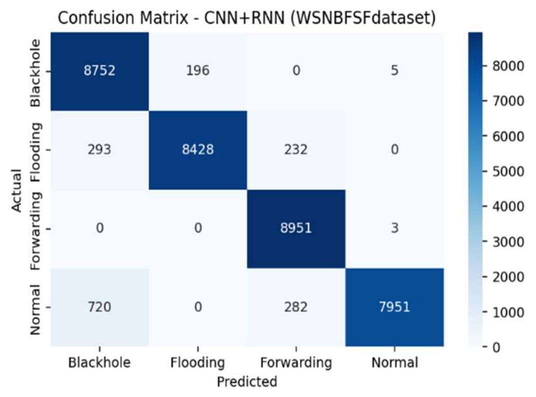
(b)



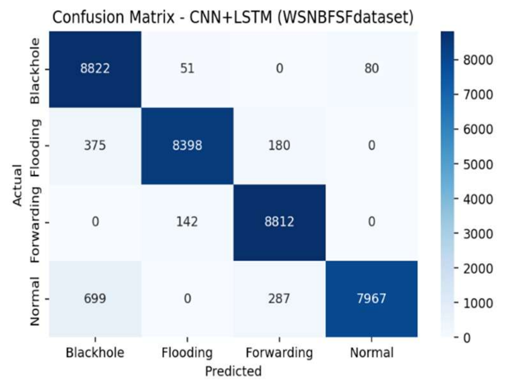
(c)



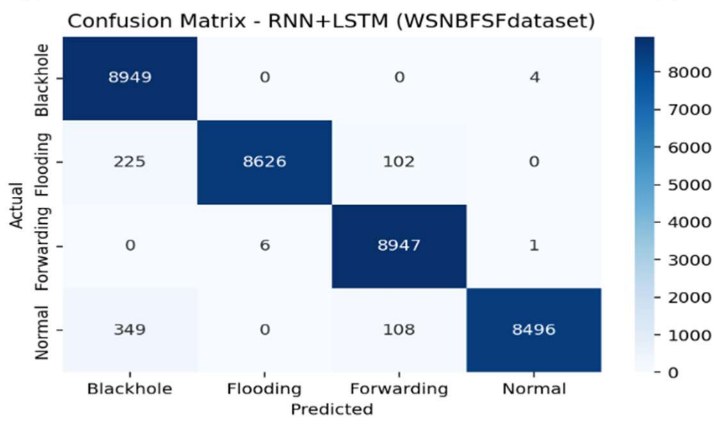
(d)



(e)



(f)



(g)

Fig. 8. Obtained confusion matrix: (a) Neural network, (b) CNN, (c) RNN, (d) LSTM, (e) CNN+RNN, (f) CNN+LSTM, (g) RNN+LSTM.

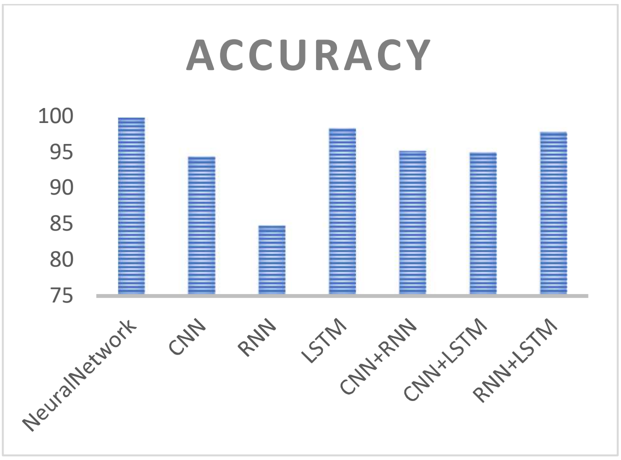


Fig. 9. Accuracy.

The precision performance implies the Neural Network model still commands the top precision as shown in Fig. 10, almost 100%, which means it positively predicted with great accuracy most of the time and was accompanied by almost no false positives. LSTM and RNN+LSTM came next at 97–98% precision level, thus carving out competent performer stature in correctly recognizing important instances. CNN+RNN, CNN+LSTM, and CNN are at the 95–96% precision level, thus quite precise but just a step below. On the lowest end of precision was the RNN model, with almost 85%, implying quite a few false positives compared to other architectures. Thus, while deep learning models are all good in terms of precision, those having memory components, especially LSTM-based models, seem to do better in reducing wrong classifications.

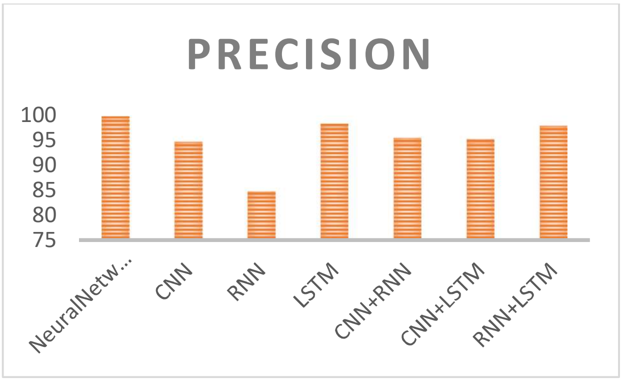


Fig. 10. Precision.

The F1−Score results shown in Fig. 11, balancing precision and recall, demonstrate that the Neural Network stands out as the most powerful one with its near-100% score, recording a superb performance both in detecting positive cases and reducing false positives. Close behind are the LSTM and RNN+LSTM models, scoring in the 97–98% range, affirming their strength in working with sequential data. The CNN+RNN, CNN+LSTM, and CNN models hold up well, scoring 95–96%, indicating consistent performance but slightly less harmony between precision and recall compared to the top performers. RNN stands behind with its F1−Score near 85%, hinting on its incapability of grasping long-term dependencies that reduce its effectiveness. Powerfully highlighting memory-enhanced architectures, particularly LSTM-based ones, which maintain a healthy balance between precision and recall, the results thus far set the yardstick for simpler recurrent or convolutional structures to follow.

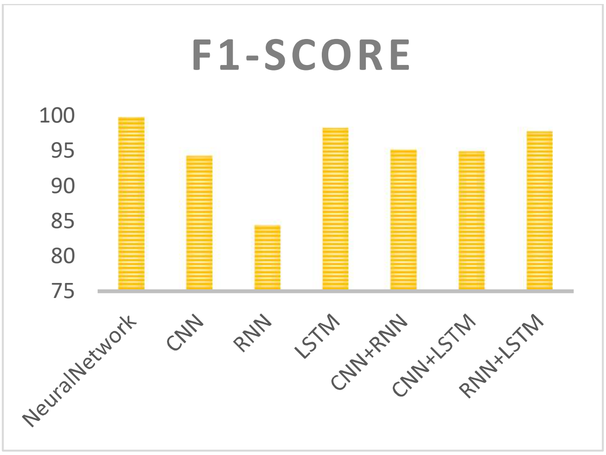


Fig. 11. F1−Score.

The recall metric shown in Fig. 12 is based on the output of various models of deep learning. Neural Networks recorded the highest recall of nearly 100%, suggesting it was able to correctly identify relevant instances. LSTM and RNN+LSTM came next with their recall at almost 98%, thus suitable for tasks requiring memory of sequential patterns. CNN classifier, alone or coupled with LSTM or RNN, clocked slightly lower recall rates of roughly 95%, which still denotes good performance but may be less effective than recurrent techniques to capture all patterns. A stand-alone RNN had the lowest recall at about 85%, suggesting it could not compete with the other architectures on this complexity level of the task. In general, LSTM-enhanced systems showed better performance than others, except for the fully connected Neural Network that defeated each implementation.

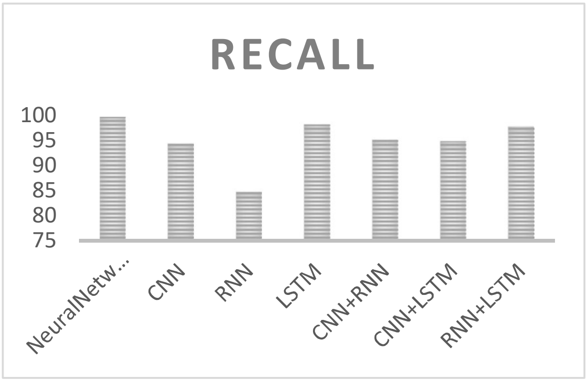


Fig. 12. recall.

The fully connected Neural Network performed nearly perfectly on all metrics, and this showed outstanding performance in learning the patterns of the datasets. LSTM and RNN+LSTM followed closely behind due to the memory mechanisms provided, which help in long-term dependencies, and CNN-based architecture did well, but not among the top. The plain RNN performed poorly due to the shortcomings of dealing with long-range sequential relationships. In general, memory augmented architectural structures were found to be the most effective when higher precision and recall were needed in the classification stage.

D. Classification Result (5-Fold Cross-Validation)

To overcome the risk of biases introduced by one train/test split and to offer more statistically rigorous evaluation of the proposed deep learning models, we performed stratified 5-fold cross-validation. The method makes every fold maintain the original class distribution of the dataset, which gives more trustworthy performance estimates. Each of the models was trained and tested five times, each fold acting as a test set once. The findings shown are the average values of these runs and its 95% Confidence Intervals (CI). Table V displays the results obtained of 5-fold cross-validation.

TABLE V. OBTAINED RESULTS (5-FOLD CROSS-VALIDATION)

Model	Accuracy	Precision	Recall	F1−Score	Time (s)
Neural Network	99.89 ± 0.04	99.89 ± 0.03	99.89 ± 0.04	99.89 ± 0.04	989.64
CNN	92.67 ± 5.38	92.92 ± 5.55	92.67 ± 5.38	92.62 ± 5.44	465.56
RNN	84.66 ± 2.15	84.72 ± 2.19	84.66 ± 2.15	84.34 ± 2.34	1201.12
LSTM	97.98 ± 2.07	98.07 ± 1.89	97.98 ± 2.07	97.98 ± 2.07	2266.99
CNN+RNN	93.86 ± 3.01	94.26 ± 2.57	93.86 ± 3.01	93.80 ± 3.12	1169.47
CNN+LSTM	95.49 ± 2.29	95.62 ± 2.29	95.49 ± 2.29	95.48 ± 2.31	1303.05
RNN+LSTM	96.92 ± 2.36	97.05 ± 2.18	96.92 ± 2.36	96.92 ± 2.37	1959.14

Table IV includes the comparison in the performance metrics (Accuracy, Precision, Recall, F1−Score) of the various DL architectures. Out of the tested architectures, the generic Neural Network (baseline) showed the best performance in all the evaluation measures, closely followed by the hybrid models, which include RNN+LSTM and CNN+LSTM. This indicates that temporal and spatial feature extraction (e.g., RNN and LSTM or CNN and LSTM) would be more suited to maximize the capability of the model to represent sequential dependencies and high-level representations, and, thus, achieve higher detection performance.

Comparing accuracy, precision, recall, and F1−Score with computational time collectively, the outcomes show that there is an evident trade-off among models under test. CNN is the fastest to calculate but only has average performance, whereas LSTM and hybrid architecture (RNN+LSTM, CNN+LSTM) have higher predictive accuracy but require much longer training. However, the neural network proves to be the most balanced of all models: it scores highest in all performance indicators but at a comparatively lower computation time than LSTM and hybrid networks. This means that the neural network is not only more efficient in reducing classification errors but is also more computationally efficient and hence is the most efficient and viable option to be used in a real-world system where classification and performance in terms of resource usage both play important factors.

E. DL Model Detail Performance Analysis

The Neural Network model proved to have the best performance in all the evaluation parameters, we have presented in detail analysis of the results of the cross-validation evaluation of the Neural Network model. Table VI shows the precision, recall and F1−Score of each attack type and the macro and micro averages. Also, we give the Area Under the Receiver Operating Characteristic Curve (ROC AUC) and Area Under the Precision-Recall Curve (PR AUC), which is a threshold-free measurement of model performance.

TABLE VI. NEURAL NETWORK PERFORMANCE METRICS FOR EACH CLASS

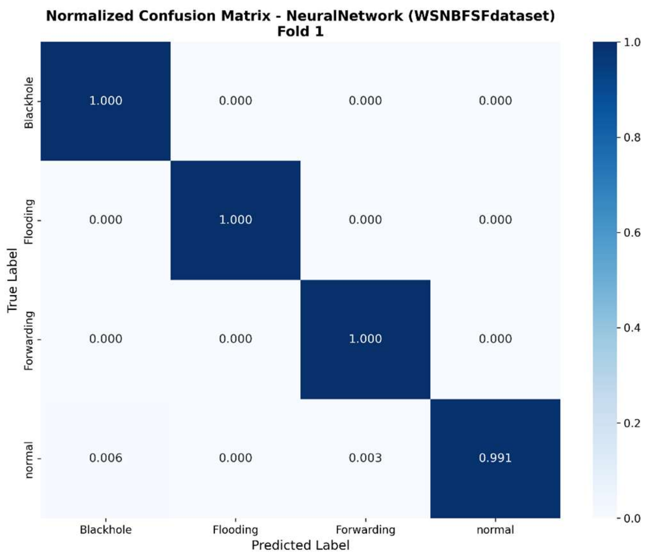
	Precision	Recall	F1−Score	ROC AUC	PR AUC
Blackhole	99.73	99.97	99.85	99.99	99.97
Flooding	100	100	100	100	100
Forwarding	99.86	99.99	99.92	100	99.97
normal	99.96	99.58	99.77	99.98	99.97
Macro Average	99.89	99.89	99.89	99.99	99.98
Micro Average	99.89	99.89	99.89	99.99	99.99

Table VI shows the performance of the Neural Network according to the classes of Blackhole, Flooding, Forwarding and Normal. The findings indicate that the model has almost perfect detection with all classes. In the Flooding category, the Neural Network obtained an ideal score of 100% in Precision, Recall, and F1−Score, with ROC AUC and PR AUC equal to 100, which means that the Neural Network did not make a single misclassification. Equally, Forwarding and Blackhole attacks were classified with very high precision and recall (all above 99.7%), leading to F1−Scores of 99.92% and 99.85 respectively. Normal class had slightly lower recall (99.58) than the other attack classes, implying that a low percentage of normal traffic was incorrectly labeled as malicious. However, its F1−Score was still high and equal to 99.77, which shows general reliable detection.

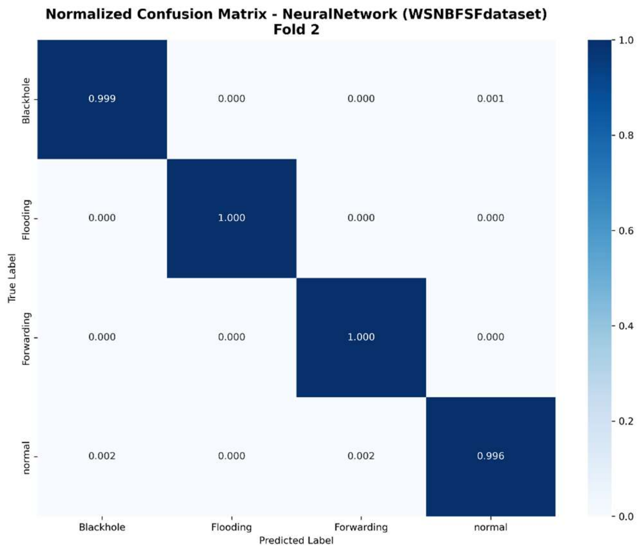
The macro- and micro-averages of the Precision, Recall, and F1−Score were all equal to 99.89, which indicates the balanced ability of the Neural Network to classify all categories without a clear bias to a specific category. Moreover, ROC AUC and PR AUC values were to 100 in most instances, which indicates that the model is very good at the discriminatory ability between positive and negative cases.

Fig. 13 indicates the normalized confusion matrix of the neural network for 5 folds. The findings support the excellent classification capability of the model with almost all samples being placed in corresponding categories correctly. All the classes are close to 1.0 indicating very high true positive values.

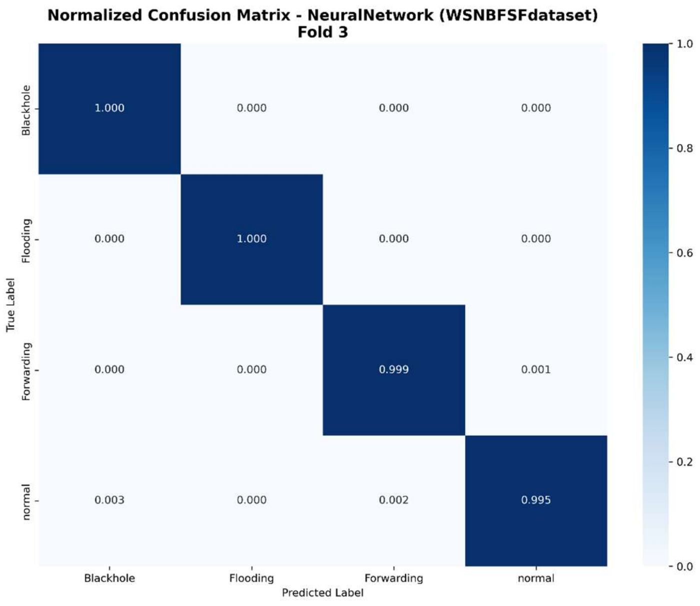
In particular, the classification of Blackhole, Flooding and Forwarding attacks was done perfectly with each having a 100% correct classification rate and no case of being mislabeled. This is an indication of the fact that the model is capable of accurately identifying volumetric and routing-based attacks. In the case of the Normal class, the neural network successfully predicted the correct classification of 99.6% and the misclassification rate of very low to Blackhole (0.3) and Forwarding (0.2) was very low. These small misclassifications can be seen as the reason behind the slightly lower recall in the previous section of the Normal category.



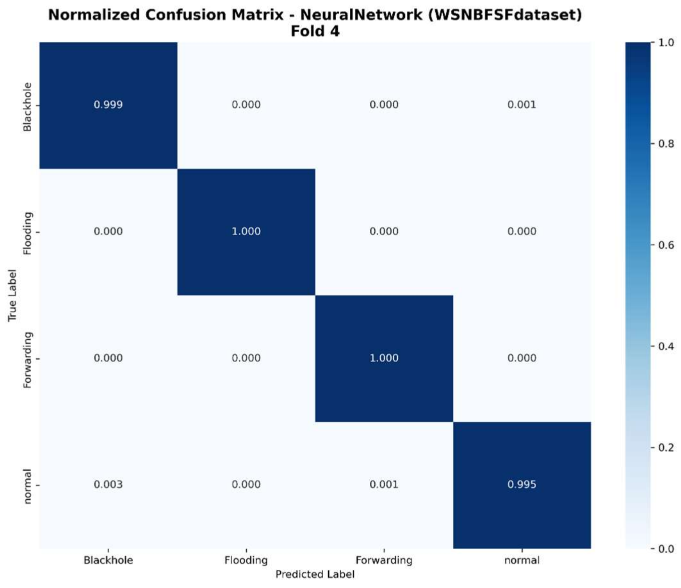
(a)



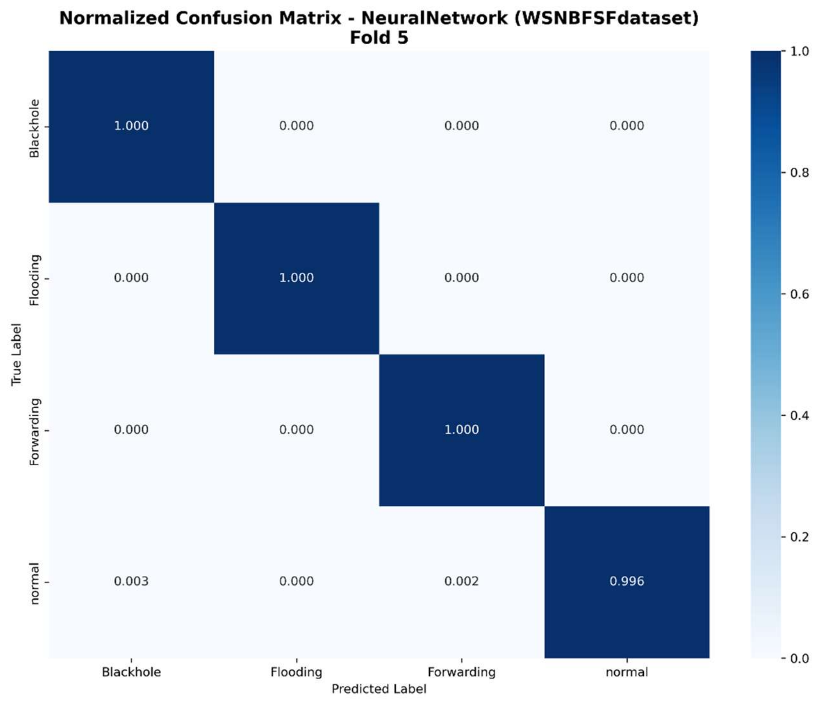
(b)



(c)



(d)



(e)

Fig. 13. Normalized confusion matrix of the neural network: (a) fold 1, (b) fold 2, (c) fold 3, (d) fold 4, (e) fold 5.

Overall, the Neural Network showed excellent results on all the evaluation measures, and it had a low misclassification error. It can be used in real-world intrusion detection systems as its capacity to maximize per-class accuracy and recall whilst demonstrating high generalization implies that it is a strong and sound candidate.

V. Conclusion and Future Work

This paper has detailed the work by performing a rigorous comparison of deep learning methods to attack detection within Wireless Sensor Networks (WSNs). As the results of the experiments, the architecture involving memory blocks, i.e., LSTM and hybrid RNN-LSTM, considered the temporal correlations of the WSN traffic with high accuracy, precision, recall, and F1−Scores. There were also good performances in fully connected neural networks, implying that they can learn complex relationships within the dataset, and CNN-based methods have delivered competitive results due to the ability to extract spatial features effectively. Standalone RNNs, in their turn, were rather ineffective, which could probably be explained by the difficulties in modeling long-term dependencies. The results reveal the possibility of deep learning in promoting WSN security through accurate and automated intrusion detection of various forms of attacks. Through models that could learn sequential and spatial patterns, WSN-based systems can attain robust and real-time threat detection, where little dependence is made on manual rule-based identification of the threats. This paper was only able to utilize a single data set, and it cannot represent the complete variation in the traffic and attack patterns of a real-life WSN, which might limit the generalizability. The experiments offline, and the expensive 3D computation requirement of certain deep learning models, could also pose challenges to applying them to resource-limited sensor nodes. Also, hyperparameter optimization was performed over a small search area, and the succession of evolving strategies of attacks can shorten long-term model performance. The further steps in the research will involve testing bigger and more varied datasets, namely on the adaptation of architectures to real-time and energy-efficiency limitation within Wireless Sensor Networks (WSNs). As the nodes of WSNs are usually constrained with energy, memory, and processing, the calculation capacity of various models, such as inference time, model size, and deployment capability, will be assessed in the future. The analysis will be used to differentiate between models that can be directly deployed onto sensor nodes and those that can be deployed to gateways or central servers. Moreover, we are going to discuss more modern methods, including parallelization, attention mechanisms, Transformer architectures, and Explainable AI to make it more efficient and interpretable. Constant learning plans will also be added in order to manage the changing attack patterns in order to ensure continued performance of detection in dynamic WSN environment.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization: Omar almomani and Areen Arabiat; methodology: Omar Almomani and Esraa Alsariera;

software: Omar Almomani; validation: Areen Arabiat: formal analysis, Areen Arabiat and Esraa Alsariera; investigation: Omar almomani and Al-Ahmed Hind; resources: Al-Ahmed Hind and Esraa Alsariera; data curation: Omar Almomani and Areen Arabiat; writing—original draft preparation, Areen Arabiat, and Omar Almomani; writing—review and editing: Al-Ahmed Hind, and Esraa Alsariera; visualization: Omar almomani and Esraa Alsariera; supervision: Omar Almomani; project administration: Areen Arabiat and Al-Ahmed Hind; funding acquisition: Omar almomani and Areen Arabiat; all authors had approved the final version.

References

[1] M. Shanmathi, A. Sonker, Z. Hussain, M. Ashraf, M. Singh, and M. Syamala, "Enhancing wireless sensor network security and efficiency with CNN-FL and NGO optimization," *Measurement: Sensors*, vol. 32, 101057, 2024.

[2] K. Ramkumar, L. H. Alzubaidi, V. Malathy, T. Venkatesh, and K. CG, "Intrusion detection system in wireless sensor networks using modified recurrent neural network with long short-term memory," presented at the 2024 International Conference on Integrated Circuits and Communication Systems (ICICACS), 2024.

[3] M. Adawy, O. Almomani, M. Tahboush, and A. Althunibat, "Integrated boolean sensing and event radius model on data aggregation in wireless sensor networks," in *Proc. 2023 International Conference on Information Technology (ICIT)*, 2023, pp. 613−619.

[4] M. A. A. Shareeda, A. M. Ali, M. A. Hammoud, Z. H. M. Kazem, and M. A. Hussein, "Secure IoT-based real-time water level monitoring system using ESP32 for critical infrastructure," *J. Cyber Secur. Risk Audit*, vol. 2, pp. 43−52, 2025.

[5] M. M. Abualhaj, M. A. Zyoud, A. Alsaaidah, A. A. Shareha, and S. A. Khatib, "Enhancing malware detection through self-union feature selection using firefly algorithm with random forest classification," *International Journal of Intelligent Engineering & Systems*, vol. 17, no. 4, 2024.

[6] M. M. Abualhaj, A. A. A. Shareha, S. N. Alkhatib, Q. Y. Shambour, and A. M. Alsaaidah, "Detecting spam using Harris Hawks optimizer as a feature selection algorithm," *Bulletin of Electrical Engineering and Informatics*, vol. 14, no. 3, pp. 2361−2369, 2025.

[7] M. Abualhaj, M. Hiari, A. Alsaaidah, and M. Al-Zyoud, "Comparative analysis of whale and Harris Hawks optimization for feature selection in intrusion detection," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 37, no. 1, p. 179, 2025.

[8] H. Alavizadeh, H. Alavizadeh, and J. J. Jaccard, "Deep Q-learning based reinforcement learning approach for network intrusion detection," *Computers*, vol. 11, no. 3, p. 41, 2022.

[9] A. Haque and H. Soliman, "A transformer-based autoencoder with isolation forest and XGBoost for malfunction and intrusion detection in wireless sensor networks for forest fire prediction," *Future Internet*, vol. 17, no. 4, p. 164, 2025.

[10] S. Konatam, S. Nalluri, M. M. Malyala, and K. K. Kandagiri, "Efficient intrusion detection model for cyber-attack mitigation in wireless sensor networks," *The International Journal of Analytical and Experimental Modal Analysis*, 2024.

[11] H. Cheng, Z. Xie, Y. Shi, and N. Xiong, "Multi-step data prediction in wireless sensor networks based on one-dimensional CNN and bidirectional LSTM," *IEEE Access*, vol. 7, pp. 117883−117896, 2019.

[12] A. Al Saaidah *et al*., "Enhancing malware detection performance: leveraging K-nearest neighbors with firefly optimization algorithm," *Multimedia Tools and Applications*, vol. 84, no. 12, pp. 10071−10094, 2025.

[13] S. N. Makhadmeh *et al*., "Recent advances in multi-objective cuckoo search algorithm, its variants and applications: SN makhadmeh et al," *Archives of Computational Methods in Engineering*, pp. 1−28, 2025.

[14] M. A. Mousa, W. Amer, M. Abualhaj, S. Albilasi, O. Nasir, and G. Samara, "Agile proactive cybercrime evidence analysis model for digital forensics," *International Arab Journal of Information Technology (IAJIT)*, vol. 22, no. 3, 2025.

[15] Y. Y. Ghadi *et al*., "Machine learning solutions for the security of wireless sensor networks: A review," *IEEE Access*, vol. 12, pp. 12699−12719, 2024.

[16] K. S. Adu-Manu, E. Amoako, and F. Engmann, "Advancements in machine learning − Enhanced green wireless sensor networks: A comprehensive survey on energy efficiency, network performance, and future directions," *Journal of Sensors*, vol. 2025, no. 1, 5242517, 2025.

[17] M. Adil, M. A. Almaiah, A. O. Alsayed, and O. Almomani, "An anonymous channel categorization scheme of edge nodes to detect jamming attacks in wireless sensor networks," *Sensors*, vol. 20, no. 8, 2311, 2020.

[18] O. Almomani, A. Alsaaidah, M. M. Abualhaj, M. A. Almaiah, A. Almomani, and S. Memon, "URL spam detection using machine learning classifiers," in *Proc. 2025 1st International Conference on Computational Intelligence Approaches and Applications (ICCIAA)*, 2025, pp. 1−6.

[19] O. Almomani, A. Alsaaidah, A. A. A. Shareha, A. Alzaqebah, and M. Almomani, "Performance evaluation of machine learning classifiers for predicting denial-of-service attack in internet of things," *International Journal of Advanced Computer Science & Applications*, vol. 15, no. 1, 2024.

[20] T. S. Delwar *et al*., "The intersection of machine learning and wireless sensor network security for cyber-attack detection: A detailed analysis," *Sensors*, vol. 24, no. 19, 6377, 2024.

[21] M. N. Khan *et al*., "Improving energy efficiency with content-based adaptive and dynamic scheduling in wireless sensor networks," *IEEE Access*, vol. 8, pp. 176495−176520, 2020.

[22] B. Almelehy, M. Ahmad, G. Nassreddine, M. Maayah, and A. Achanta, "Analytical analysis of cyber threats and defense mechanisms for web application security," *Journal of Cyber Security and Risk Auditing*, no. 3, pp. 57−76, 2025.

[23] G. Lippi, M. Aljawarneh, Q. A. Na'amneh, R. Hazaymih, and L. D. Dhomeja, "Security and privacy challenges and solutions in autonomous driving systems: A comprehensive review," *Journal of Cyber Security and Risk Auditing*, no. 3, pp. 23−41, 2025.

[24] F. A. Quayed, Z. Ahmad, and M. Humayun, "A situation based predictive approach for cybersecurity intrusion detection and prevention using machine learning and deep learning algorithms in wireless sensor networks of industry 4.0," *IEEE Access*, vol. 12, pp. 34800−34819, 2024.

[25] A. Davarasan, J. Samual, K. Palansundram, and A. Ali, "A comprehensive review of machine learning approaches for android malware detection," *Journal of Cyber Security and Risk Auditing*, no. 1, pp. 38−60, 2024.

[26] C. R. Morales, F. R. Sousa, V. Brusamarello, and N. C. Fernandes, "Evaluation of deep learning methods in a dual prediction scheme to reduce transmission data in a WSN," *Sensors*, vol. 21, no. 21, 7375, 2021.

[27] G. A. Sukkar and S. A. Sharaeh, "Enhancing security in wireless sensor networks: A machine learning-based DoS attack detection," *Engineering, Technology and Applied Science Research*, vol. 15, no. 1, pp. 19712−19719, 2025.

[28] P. Phalaagae, A. M. Zungeru, A. Yahya, B. Sigweni, and S. Rajalakshmi, "A hybrid CNN-LSTM model with attention mechanism for improved intrusion detection in wireless IoT sensor networks," *IEEE Access*, 2025.

[29] A. Tolba, "Hybrid LSTM-random forest for intrusion detection in wireless sensor networks: Enhanced DoS classification with explainable AI," *International Journal of Computers and Informatics*, vol. 2, pp. 39−52, 2024.

[30] A. I. Altameemi, S. J. Mohammed, Z. Q. Mohammed, Q. K. Kadhim, and S. T. Ahmed, "Enhanced SVM and RNN classifier for cyberattacks detection in underwater wireless sensor networks," *International Journal of Safety and Security Engineering*, vol. 14, no. 5, 2024.

[31] G. Reddy *et al*., "An intrusion detection using machine learning algorithm Multi-Layer Perceptron (MlP): A classification enhancement in Wireless Sensor Network (WSN)," *International Journal of Recent Innovations in Computing and Communication*, vol. 10, no. 2, pp. 139−145, 2022.

[32] S. Salmi and L. Oughdir, "CNN-LSTM based approach for dos attacks detection in wireless sensor networks," *International*

Journal of Advanced Computer Science and Applications, vol. 13, no. 4, 2022.

[33] S. Sivakumar, M. M. Theresa, K. Sudha, and K. Sangeethalakshmi, "Secure wireless sensor networks: A weighted K-NN and RNN-based approach for attack detection and localization," *IETE Journal of Research*, vol. 71, no. 3, pp. 864−875, 2025.

[34] A. R. A. Moundounga and H. Satori, "Stochastic machine learning based attacks detection system in wireless sensor networks," *Journal of Network and Systems Management*, vol. 32, no. 1, p. 17, 2024.

[35] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "CNN-LSTM: hybrid deep neural network for network intrusion detection system," *IEEE Access*, vol. 10, pp. 99837−99849, 2022.

[36] WSNBFSF Dataset. [Online]. Available: https://www.kaggle.com/datasets/celilokur/wsnbfsfdataset

[37] T. M. Nguyen, H. H. P. Vo, and M. Yoo, "Enhancing intrusion detection in wireless sensor networks using a GSWO-CatBoost approach," *Sensors*, vol. 24, no. 11, 3339, 2024.

[38] T. Zhukabayeva, A. Pervez, Y. Mardenov, M. Othman, N. Karabayev, and Z. Ahmad, "A traffic analysis and node categorization-aware machine learning-integrated framework for cybersecurity intrusion detection and prevention of WSNs in smart grids," *IEEE Access*, vol. 12, pp. 91715−91733, 2024.

[39] A. Arabiat and M. Altayeb, "Driving behavior analytics: An intelligent system based on machine learning and data mining techniques," *Bulletin of Electrical Engineering and Informatics*, vol. 14, no. 3, pp. 2055−2065, 2025.

[40] J. Asian, M. D. Rosita, and T. Mantoro, "Sentiment analysis for the Brazilian anesthesiologist using multi-layer perceptron classifier and random forest methods," *Jurnal Online Informatika*, vol. 7, no. 1, pp. 132−141, 2022.

[41] A. Arabiat and M. Altayeb, "Enhancing internet of things security: Evaluating machine learning classifiers for attack prediction," *International Journal of Electrical & Computer Engineering*, vol. 14, no. 5, 2024.

[42] Y. Wang, J. Tao, and L. Zhao, "Incremental multi-step learning MLP model for online soft sensor modeling," *Sensors*, vol. 25, no. 14, 4303, 2025.

[43] B. Rusyn *et al.*, "Rethinking deep CNN training: A novel approach for quality-aware dataset optimization," *IEEE Access*, vol. 12, pp. 137427−137438, 2024.

[44] C. Eang and S. Lee, "Predictive maintenance and fault detection for motor drive control systems in industrial robots using CNN-RNN-based observers," *Sensors*, vol. 25, no. 1, p. 25, 2024.

[45] H. Ding, H. Hou, L. Wang, X. Cui, W. Yu, and D. I. Wilson, "Application of convolutional neural networks and recurrent neural networks in food safety," *Foods*, vol. 14, no. 2, p. 247, 2025.

[46] A. Duraj, P. S. Szczepaniak, and A. Sadok, "Detection of anomalies in data streams using the LSTM-CNN model," *Sensors*, vol. 25, no. 5, 1610, 2025.

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735−1780, 1997.

[48] M. A. Almedires, A. Elkhalil, and M. Amin, "Adversarial attack detection in industrial control systems using LSTM-based intrusion detection and black-box defense strategies," *Journal of Cyber Security and Risk Auditing*, no. 3, pp. 4−22, 2025.

[49] I. D. Mienye, T. G. Swart, and G. Obaido, "Recurrent neural networks: A comprehensive review of architectures, variants, and applications," *Information*, vol. 15, no. 9, p. 517, 2024.

[50] Q. Guo, Z. He, and Z. Wang, "Assessing the effectiveness of long short-term memory and artificial neural network in predicting daily ozone concentrations in Liaocheng city," *Scientific Reports*, vol. 15, no. 1, 6798, 2025.

[51] A. M. A. Ghamdi and M. M. Alansari, "Enhancing IoT security: A comparative study of CNN and RNN-based anomaly detection using the CICIoT2023 dataset," *IAENG International Journal of Computer Science*, vol. 52, no. 5, 2025.

[52] S. Kumar *et al.*, "Leveraging neural networks (RNN_LSTM) in enhancing energy efficiency and network lifetime in WSNs," *International Journal of Information Technology*, pp. 1−6, 2025.

[53] O. Almomani, M. A. Almaiah, M. Madi, A. Alsaaidah, M. A. Almomani, and S. Smadi, "Reconnaissance attack detection via boosting machine learning classifiers," *AIP Conference Proceedings*, vol. 2979, no. 1, 2023.

[54] O. Almomani, "A hybrid model using bio-inspired metaheuristic algorithms for network intrusion detection system," *Computers, Materials and Continua*, vol. 68, no. 1, 2021.

[55] D. M. Kalász, I. Bodnár, and M. Jobbágy, "An overview of CNN-based image analysis in solar cells, photovoltaic modules, and power plants," *Applied Sciences*, vol. 15, no. 10, 5511, 2025.

[56] A. M. Arabiat, "Intelligent model for detecting GAN-generated images based on multi-classifier and advanced data mining techniques," *International Journal of Electrical and Electronic Engineering and Telecommunications*, vol. 14, no. 3, 2025.

[57] X. Zhang *et al.*, "Data-driven loan default prediction: A machine learning approach for enhancing business process management," *Systems*, vol. 13, no. 7, p. 581, 2025.

[58] M. Altayeb and A. Arabiat, "A sustainable system for predicting appliance energy consumption based on machine learning," *Journal of Environmental Management*, vol. 382, 125434, 2025.

[59] A. Alizargar, Y. L. Chang, and T. H. Tan, "Performance comparison of machine learning approaches on hepatitis C prediction employing data mining techniques," *Bioengineering*, vol. 10, no. 4, p. 481, 2023.

[60] A. Arabiat, M. Hassan, and O. Almomani, "WEKA-based machine learning for traffic congestion prediction in Amman city," *Int. J. Artif Intell ISSN*, vol. 2252, no. 8938, 4423, 2023.

[61] A. Taner, Y. B. Öztekin, and H. Duran, "Performance analysis of deep learning CNN models for variety classification in hazelnut," *Sustainability*, vol. 13, no. 12, 6527, 2021.

[62] N. G. Sorboni, J. Wang, and M. R. Najafi, "Fusion of google street view, LiDAR, and orthophoto classifications using ranking classes based on F1−Score for building land-use type detection," *Remote Sensing*, vol. 16, no. 11, 2024.